📚 What is Recursion ?

It is a method in which a function calls itself usually with smaller inputs.

`f(n) → f(n-1) → f(n-2) → f(n-3)...`

When a function `A()` calls a function `B()`, it is called **function calling**.

When a function `A()` calls the same function again, it is called a **function calling itself**.

❓ Where Do We Need Recursion?

Choices and decisions are given.

🛑 When to Stop?

`f(n) → f(n-1) → f(n-2) → f(n-3) → ... ?`

🔁 Base Condition

The point where further function calls stop.

Usually based on:

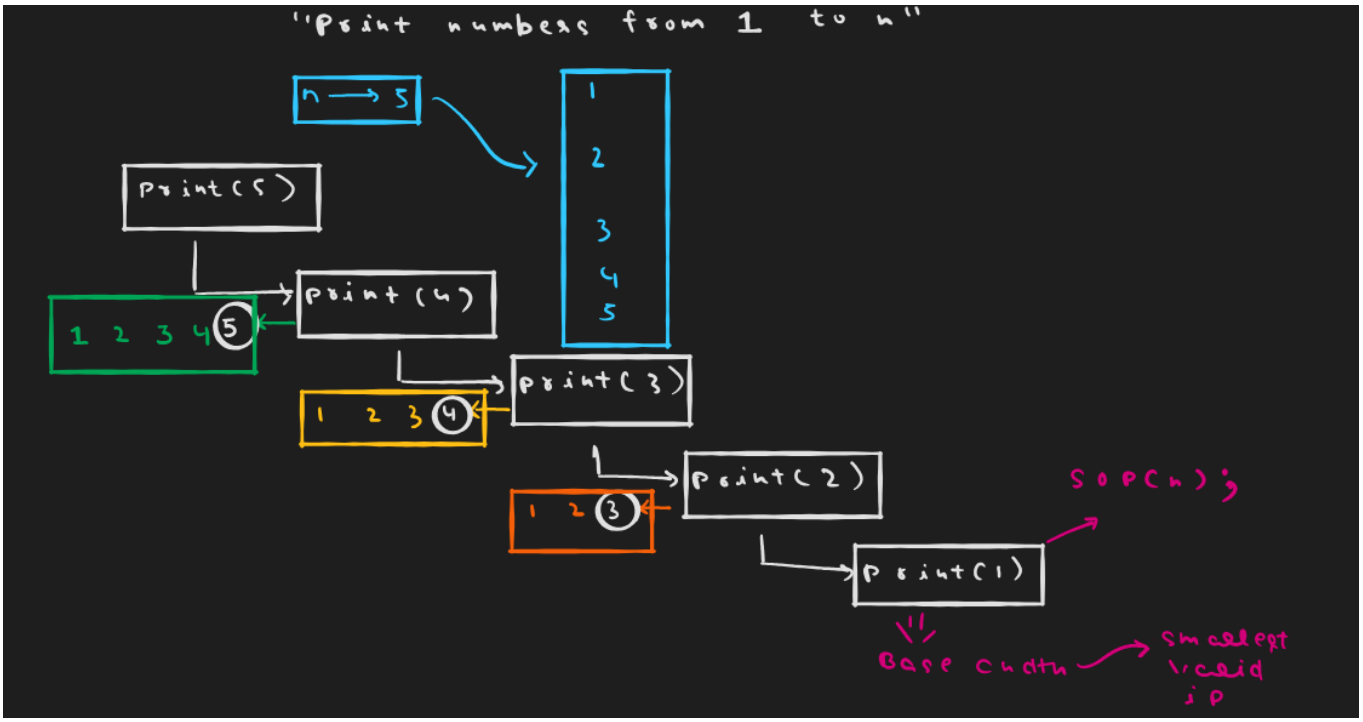
1. **Smallest valid input**
2. **Largest invalid input**

"Recursion works like magic."

`f(n) → f(n-1)`

Trust that `f(n-1)` will always return a valid output.

🔢 Print Numbers from 1 to n Using Recursion



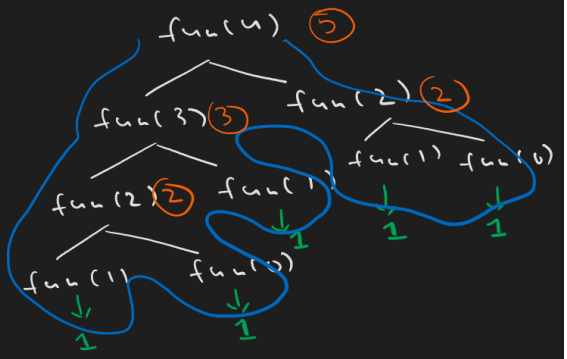

```java
public class Main {
	public static void main(String[] args) {
		int n = 5;
```

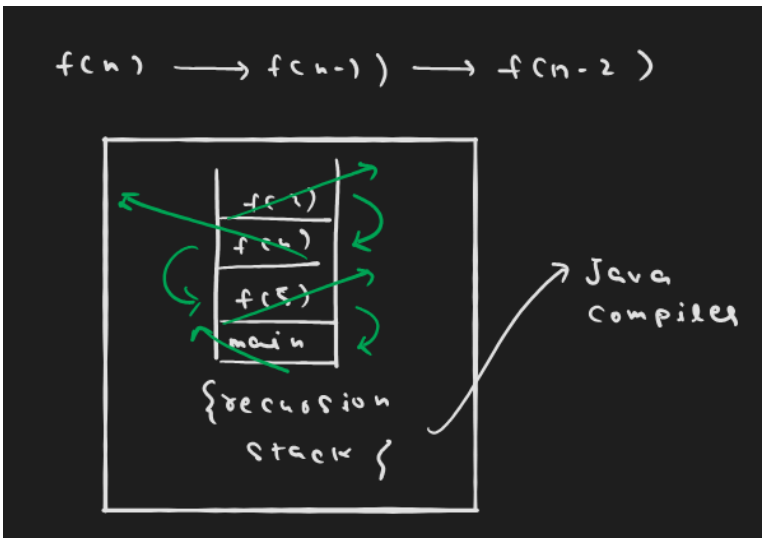
```
            printIncreasing(n);
      }

      public static void printIncreasing(int n) {
            if (n == 0) {
                  return;
            }
            printIncreasing(n - 1);
            System.out.println(n);
      }
}
```

What is a Recursive Tree ?

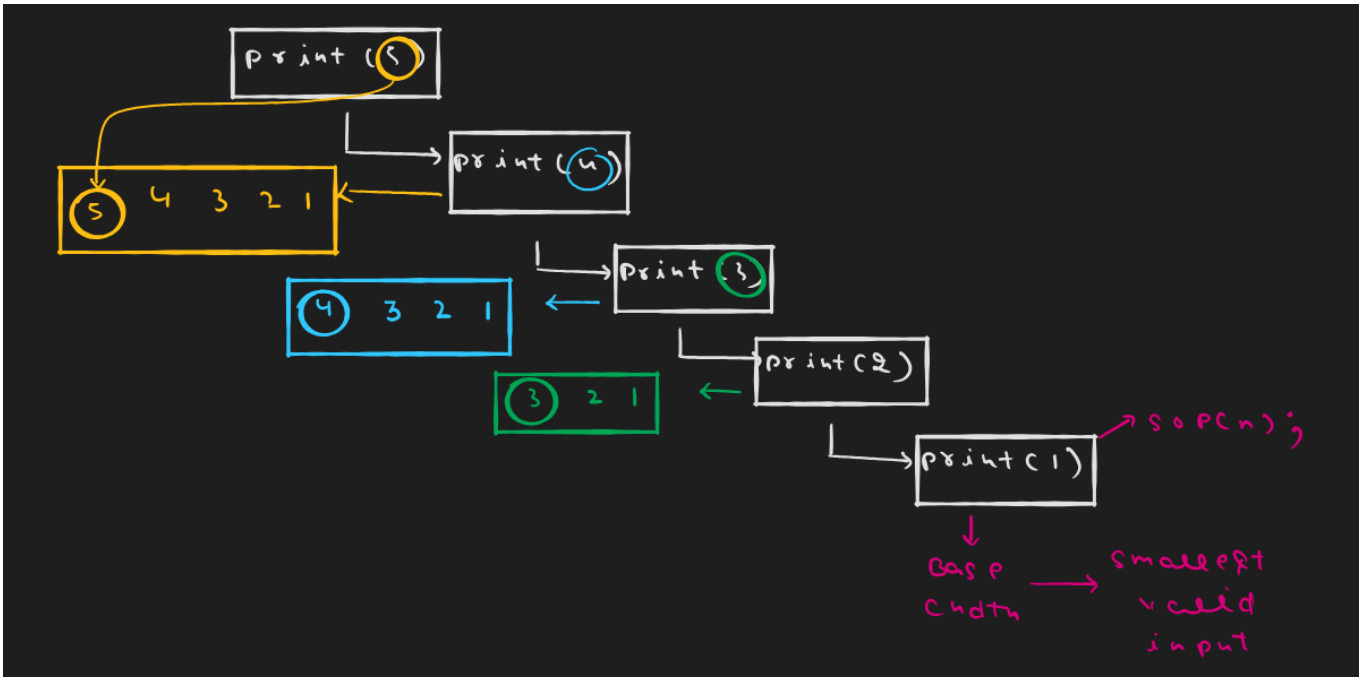
It visualizes the code flow of a recursive code.

```
def fun(n):
    if n==1 or n==0:
        return 1
    else:
        return fun(n-1)+fun(n-2)

print(fun(4))
```



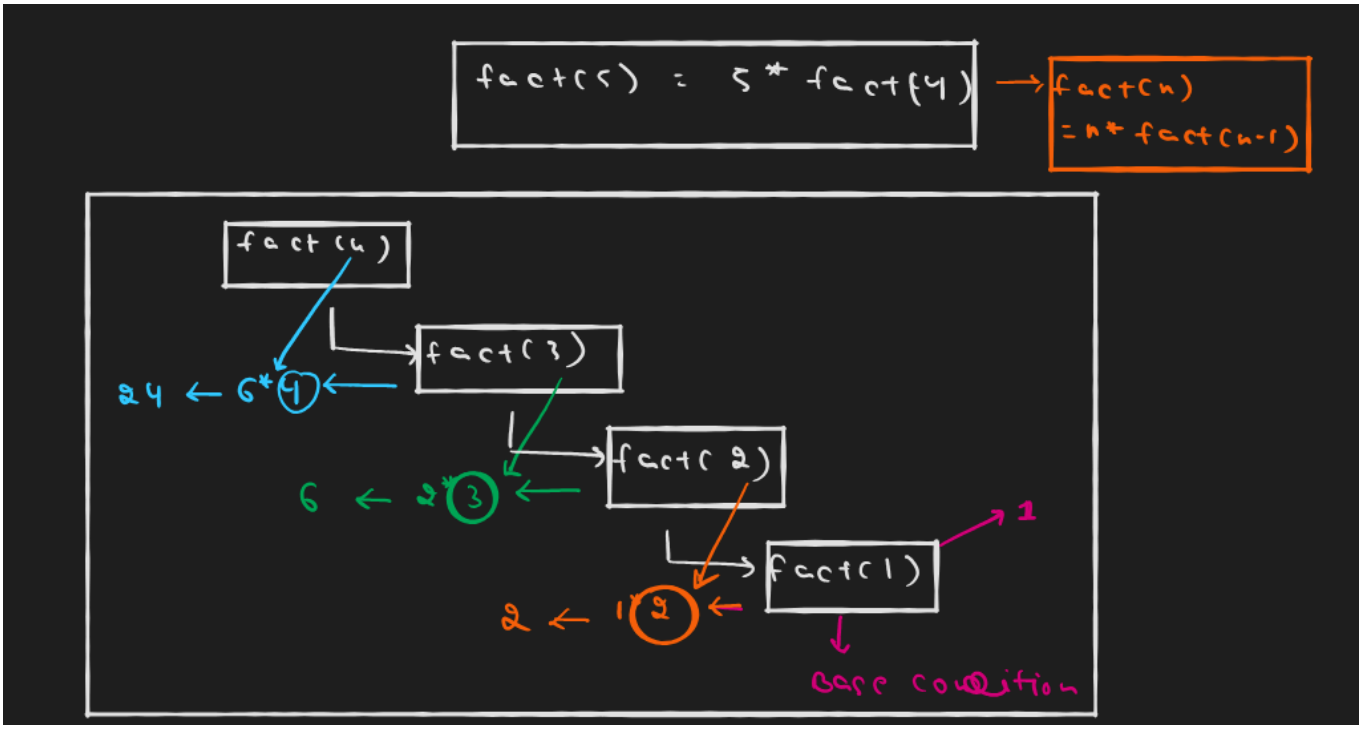


🔢 Print Numbers from n to 1 Using Recursion




```java
public class Main {
	public static void main(String[] args) {
		int n = 5;
		printDecreasing(n);
	}

	public static void printDecreasing(int n) {
		if (n == 0) {
			return;
		}
		System.out.println(n);
		printDecreasing(n - 1);
	}
}
```

➗ Factorial of a Number Using Recursion




```java
public class Main {
	public static void main(String[] args) {
		int n = 5;
		int factorial = fact(n);
	  System.out.println(factorial);
	}

	public static int fact(int n){
	  if(n==1)
		return 1;
	  return n*fact(n-1);
    }
}
```

⚡ Power of a Number Using Recursion

```java
public class Main {
	public static void main(String[] args) {
		int a = 3;
		int n = 4;
		System.out.println(power(a, n));
	}

	public static int power(int a, int n) {
		if (n == 0) {
			return 1;
		}
		int m = power(a, n - 1);
		return m * a;
	}
}
```

🔍 First Occurrence of a Number in an Array Using Recursion

```java
public class Main {
	public static void main(String[] args) {
		int[] arr = { 1, 2, 5, 4, 3, 4, 7, 4, 3, 6 };
		int item = 4;
		System.out.println(index(arr, item, 0));
	}

	public static int index(int[] arr, int item, int i) {
		if (i == arr.length) {
			return -1;
		}
		if (arr[i] == item) {
			return i;
		}
		return index(arr, item, i + 1);
	}
}
```

✅ Check if an Array is Sorted Using Recursion

```java
public class Main {
	public static void main(String[] args) {
		int[] arr = { 5, 7, 8, 11, 15 };
		System.out.println(isSorted(arr, 0));
	}

	public static boolean isSorted(int[] arr, int i) {
		if (i == arr.length - 1) {
			return true;
		}
		if (arr[i] > arr[i + 1]) {
			return false;
		}
		return isSorted(arr, i + 1);
	}
}
```